



Pós-graduação Lato Sensu em
Ciência de Dados e Big Data
Trabalho de Conclusão de Curso

PUC Minas
Virtual

MODELO PREDITIVO DE VARIAÇÃO DA
CRIPTOMOEDA BITCOIN

Aluno: **Jean Carlos Sousa Silva**

Belo Horizonte
2022



Introdução

- O Mercado de Cripto Moedas
- Risco de investimento
- Uso da tecnologia para facilitar a vida do investidor
- Criação de modelos matemáticos para previsão

Contextualização

- O Nascimento do Bitcoin
- Segurança do Modelo
- Aceitação no Mercado Global
- Novo Padrão Monetário Mundial

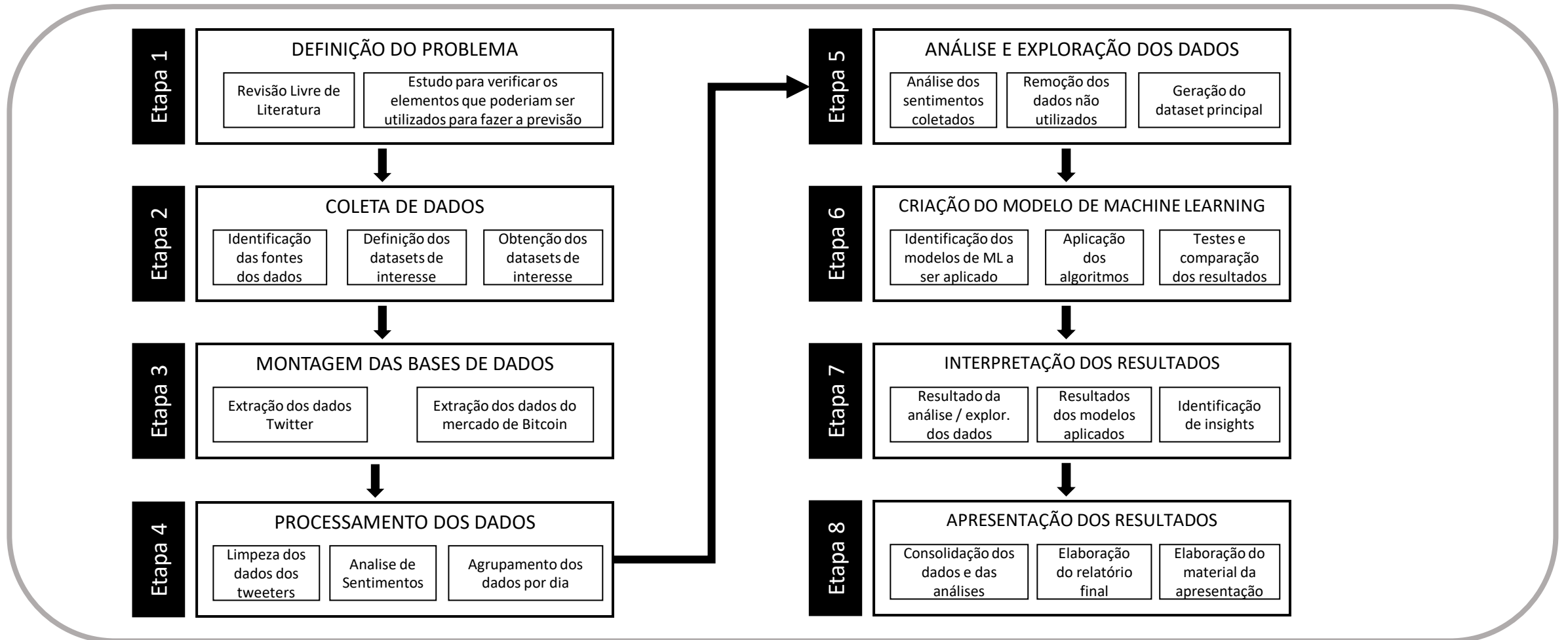
Problema Proposto

- Este trabalho busca responder as seguintes questões:
 - É possível prever a variação do Bitcoin?
 - O que é discutido nas redes sociais pode ajudar nesta previsão?
 - Como utilizar as discussões sobre o tema para melhorar as previsões?

Estratificação do Problema

W	Resposta
Porque?	Grande variação nos preços do Bitcoin
Quem?	Informações sobre variação da moeda no mercado e os tweets do que é discutido sobre o Bitcoin
O que?	Criar uma ferramenta para auxiliar a investir em Bitcoin
Onde?	Para o investidor no mercado global
Quando?	O período analisado foi o ano de 2021

Metodologia



Metodologia

- Para execução deste trabalho foram utilizados:
 - Linguagem de programação Python na versão 3.9.4, pelo Visual Studio Code para a coleta de dados, montagem das bases de dados, processamento dos dados e para análise, exploração dos dados, criação do modelo de machine learning e interpretação dos dados foi utilizado o Google Colab.
 - Foi utilizado o Microsoft Excel (para geração das tabelas), o Microsoft Word (para geração do relatório final) e o Microsoft Powerpoint (para criar o material de apresentação)

Bibliotecas do Python

- Para coleta, montagem das bases, processamento, análise e exploração dos dados:

```
#Bibliotecas gerais
from ast import keyword
from distutils.command.clean import clean
from itertools import count
from datetime import datetime
from textblob import TextBlob
from PIL import Image
from wordcloud import WordCloud, STOPWORDS
import tweepy
import re
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import warnings
import math
from vaderSentiment.vaderSentiment import SentimentIntensityAnalyzer
```

- Tratamento de arquivos:

```
import os
import glob
import pandas as pd
```

- Criação do modelo de machine learning:

```
#treinamento da rede neural
import pandas as pd
import datetime
import random
import time
import numpy as np
import collections
import pickle
from sklearn.neural_network import MLPClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
```


Coleta de Dados

- Dados do Twitter
 - Captura dos tweets dos usuários envolvendo os termos do Bitcoin
 - Dados publicados na plataforma do ano de 2021
 - Captados 1.000 últimos tweets de cada dia
- Dados do mercado de Bitcoin
 - Informações sobre o mercado de Bitcoin com os valores de abertura, fechamento, volume de transações, maior e menor valor transacionado do dia.

Coleta de Dados

- Obtenção dos Dados do Twitter

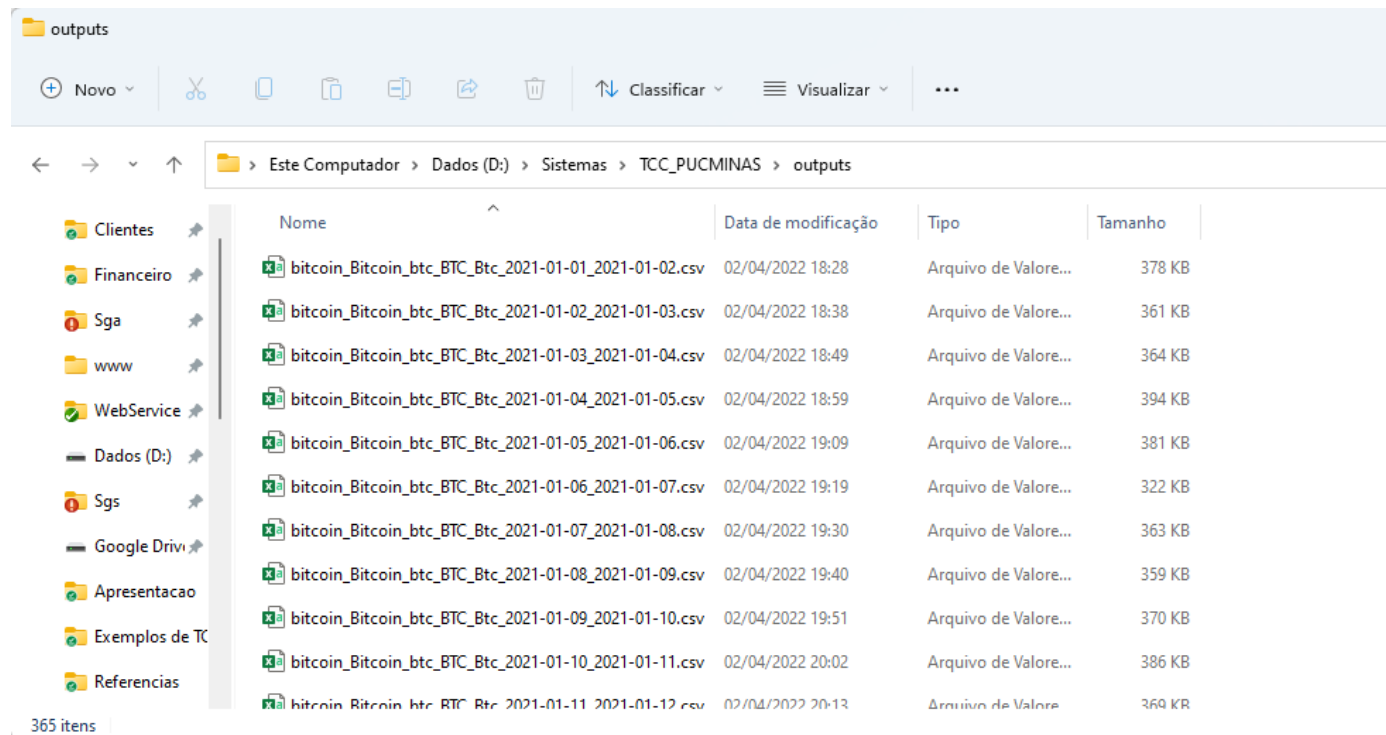
```
from calendar import month
from datetime import date, timedelta
from multiprocessing.connection import wait
import pandas as pd
from Sweets.scweet import scrape
from textblob import TextBlob
import datetime as dt
import time

def scrap_tweets(start_date, end_date):
    start_time = time.time()

    with open('logs.txt', 'a+') as f:
        f.write(f"Starting on...\n")
        tweets = scrape(words=['bitcoin', 'Bitcoin', 'btc', 'BTC', 'Btc'], since=start_date.strftime('%Y-%m-%d'),
                        until=end_date.strftime('%Y-%m-%d'), from_account=None, interval=1, resume=False, filter_replies=False,
                        proximity=False, lang='en', display_type='Latest', limit=1000)
        f.write(f"---- Found {len(tweets)} tweets. ----\n")
        f.write(f"---- Took {time.time() - start_time} seconds ----\n")
```

Coleta de Dados

- 364 arquivos de dados do Twitter gerados



Coleta de Dados

- Obtenção do Mercado de Bitcoin

```
import requests
import time
import datetime

class BitcoinRequest:
    def __init__(self, api_key):
        self.api_key = api_key
        self.BASE_URL = 'https://min-api.cryptocompare.com/data/v2'

    def __get_data(self, response):
        return response.json()['Data']['Data']

    def get_dialy_data(self, fsym='BTC', tsym='USD', aggregate=1, limit=365):
        """
        Get values of bitcoin market
        """
        response = requests.get(f'{self.BASE_URL}/histoday?fsym={fsym}&tsym={tsym}&aggregate={aggregate}&limit={limit}&api_key={self.api_key}')
        return self.__get_data(response)
```

Limpeza/Tratamento dos Dados

- Remoção de informações para manter nos textos dos tweets capturados “texto útil” usando RegEx.
- Remoção colunas com informação não pertinentes para este trabalho
- Captura da polaridade dos sentimentos dos textos capturados com a ferramenta VADER.
- Agrupamento das informações por dia

Limpeza/Tratamento dos Dados

Limpeza dos tweets

```
def __clean(self, text):  
    '''  
    Cleaning a text  
    '''  
    clean_text = re.sub(r'RT+', '', text)  
    clean_text = re.sub(r'@\S+', '', clean_text)  
    clean_text = re.sub(r'https?\S+', '', clean_text)  
    clean_text = clean_text.replace('\n', ' ')  
    clean_text = clean_text.replace('Em resposta', '')  
    clean_text = clean_text.replace('Mostrar esta', '')  
  
    return clean_text
```

Total de tweets

Tweets Capturados	
Total de Tweets	Média de Tweets por dia
362.708	1.000

Análise de sentimentos VADER

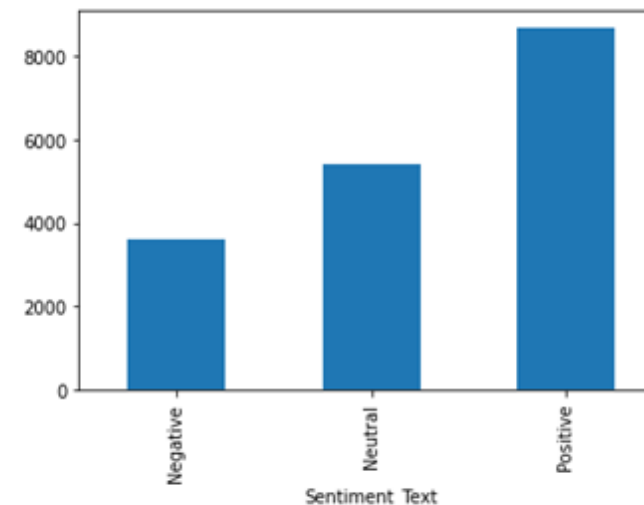
```
def sentiment_polarity_from_vader(self, tweets_text_list):  
    '''  
    Sentimental analyse  
    '''  
    tweets_sentiments_list = []  
    sentiment_text = ''  
  
    for tweet in tweets_text_list:  
        polarity = self.vaderAnalyzer.polarity_scores(tweet)  
  
        if polarity['compound'] >= 0.05:  
            sentiment_text = 'Positive'  
        elif polarity['compound'] <= -0.05:  
            sentiment_text = 'Negative'  
        else:  
            sentiment_text = 'Neutral'  
        tweets_sentiment = {  
            'Sentiment_Compound':polarity['compound'],  
            'Sentiment_Neutral':polarity['neu'],  
            'Sentiment_Negative':polarity['neg'],  
            'Sentiment_Positive':polarity['pos'],  
            'Sentiment_Text':sentiment_text  
        }  
        tweets_sentiments_list.append(tweets_sentiment)  
  
    return tweets_sentiments_list
```

Análise e exploração dos Dados

Nuvem de palavras dos textos dos tweets



Sentimentos dos textos dos tweets



<Figure size 432x288 with 0 Axes>

Análise e exploração dos Dados

Agrupamento dos dados por dia

```
from datetime import date, timedelta
import time

def daterange(start_date, end_date):
    for n in range(int((end_date - start_date).days)):
        yield start_date + timedelta(n)

def groupTweetsByDay(tweets):
    # Grouping tweets by day
    start_date = date(2021, 1, 1)
    end_date = date(2022, 1, 1)
    dtypes = np.dtype([('date', str), ('neu_mean', float), ('neg_mean', float), ('pos_mean', float), ('comp_mean', float), ('pol_mean', float), ('qtd_day', int)])
    new_df = pd.DataFrame(np.empty(0, dtype=dtypes))

    for d in daterange(start_date, end_date):
        sum_neu = tweets[(tweets.Date==d.strftime("%Y-%m-%d"))].sum()['Sentiment_Neutral']
        sum_neg = tweets[(tweets.Date==d.strftime("%Y-%m-%d"))].sum()['Sentiment_Negative']
        sum_pos = tweets[(tweets.Date==d.strftime("%Y-%m-%d"))].sum()['Sentiment_Positive']
        sum_comp = tweets[(tweets.Date==d.strftime("%Y-%m-%d"))].sum()['Sentiment_Compound']
        total_tweets_of_day = tweets[(tweets.Date==d.strftime("%Y-%m-%d"))].count()['Date']
        new_row = {
            'date': d.strftime("%Y-%m-%d"),
            'neu_mean': sum_neu/total_tweets_of_day,
            'neg_mean': sum_neg/total_tweets_of_day,
            'pos_mean': sum_pos/total_tweets_of_day,
            'comp_mean': sum_comp/total_tweets_of_day,
            'pol_mean': math.sqrt(sum_pos/total_tweets_of_day * sum_neg/total_tweets_of_day),
            'qtd_day': total_tweets_of_day
        }
        new_df = new_df.append(new_row, ignore_index=True)

    return new_df
```

Dataset com análise de sentimentos dos tweets

	date	neu_mean	neg_mean	pos_mean	comp_mean	pol_mean	qtd_day
0	2021-01-01	0.854263	0.045843	0.099883	0.182852	0.067668	1000
1	2021-01-02	0.865478	0.044039	0.090482	0.155658	0.063125	1000
2	2021-01-03	0.862065	0.041362	0.096569	0.169264	0.063200	1000
3	2021-01-04	0.864370	0.040225	0.095397	0.202557	0.061946	1000
4	2021-01-05	0.864378	0.036116	0.099508	0.216036	0.059949	1000
...
360	2021-12-27	0.859985	0.046149	0.093852	0.150913	0.065812	1000
361	2021-12-28	0.855977	0.051500	0.092514	0.145640	0.069025	1000
362	2021-12-29	0.858153	0.056188	0.085661	0.115104	0.069377	1000
363	2021-12-30	0.853271	0.047745	0.098975	0.172850	0.068743	1000
364	2021-12-31	0.839904	0.044251	0.115844	0.218459	0.071598	1000

365 rows x 7 columns

Análise e exploração dos Dados

Adicionando os dados do mercado de Bitcoin

```
apy_key = "582a4eea85fa4058d78c6994e9f7af704768d5d5eb93e10dee801962c3695823"

bitcoinRequest = BitcoinRequest(api_key=apy_key)
data_inicial = datetime.datetime(2020, 12, 31)
data_final = datetime.datetime.now()
days = abs(data_inicial-data_final).days
jsonData = bitcoinRequest.get_dialy_data(fsym='BTC', tsym='USD', aggregate=1, limit=days)

# add informations about bitcoin
bt_close = []
bt_open = []
bt_high = []
bt_low = []
bt_volumeto = []
bt_target = []
for item in new_df.itertuples(index=False):
    timestamp = time.mktime(datetime.datetime.strptime(item.date, "%Y-%m-%d").timetuple())
    index = bitcoinRequest.get_index(jsonData, timestamp)
    if index != None:
        bt_close.append(jsonData[index-1]['close'])
        bt_open.append(jsonData[index]['open'])
        bt_high.append(jsonData[index]['high'])
        bt_low.append(jsonData[index]['low'])
        bt_volumeto.append(jsonData[index]['volumeto'])
        bt_target.append(bitcoinRequest.get_target_function(jsonData[index-1]['close'], jsonData[index]['close']))

new_df['bt_close'] = bt_close
new_df['bt_open'] = bt_open
new_df['bt_high'] = bt_high
new_df['bt_low'] = bt_low
new_df['bt_volumeto'] = bt_volumeto
new_df['bt_target'] = bt_target
```

Análise e exploração dos Dados

Dataset final para submeter aos modelos

	date	neu_mean	neg_mean	pos_mean	comp_mean	pol_mean	qtd_day	bt_close	bt_open	bt_high	bt_low	bt_volumeto	bt_target
0	2021-01-01	0.854263	0.045843	0.099883	0.182852	0.067668	1000	28972.40	28972.40	29666.33	28748.21	1.487306e+09	1
1	2021-01-02	0.865478	0.044039	0.090482	0.155658	0.063125	1000	29388.94	29388.94	33257.29	29036.26	3.750441e+09	1
2	2021-01-03	0.862065	0.041362	0.096569	0.169264	0.063200	1000	32203.64	32203.64	34789.34	32010.59	3.132759e+09	1
3	2021-01-04	0.864370	0.040225	0.095397	0.202557	0.061946	1000	33063.48	33063.48	33622.70	28493.29	3.901979e+09	-1
4	2021-01-05	0.864378	0.036116	0.099508	0.216036	0.059949	1000	32030.55	32030.55	34487.44	29985.29	3.037657e+09	1
...
360	2021-12-27	0.859985	0.046149	0.093852	0.150913	0.065812	1000	50790.88	50790.88	52079.46	50481.26	1.120188e+09	-1
361	2021-12-28	0.855977	0.051500	0.092514	0.145640	0.069025	1000	50714.73	50714.73	50718.53	47317.78	1.985905e+09	-1
362	2021-12-29	0.858153	0.056188	0.085661	0.115104	0.069377	1000	47536.39	47536.39	48145.34	46107.79	1.599557e+09	-1
363	2021-12-30	0.853271	0.047745	0.098975	0.172850	0.068743	1000	46471.70	46471.70	47917.89	45964.28	1.852207e+09	1
364	2021-12-31	0.839904	0.044251	0.115844	0.218459	0.071598	1000	47129.66	47129.66	48573.22	45665.44	1.636014e+09	-1

365 rows × 13 columns

```
new_df.to_csv("matriz_resultado_redes_neurais.csv", index=False)
```

Criação do modelo de Machine Learning

Definição da matriz para o treinamento e alvo

```
df_train_matrix = pd.read_csv('matriz_resultado_rede_neural.csv', header=0, dtype={'date': 'str', 'neu_mean': 'float', 'neg_mean': 'float',  
                                         'pos_mean': 'float', 'comp_mean': 'float', 'pol_mean': 'float',  
                                         'bt_close': 'float', 'bt_open': 'float', 'bt_high': 'float', 'bt_low': 'float',  
                                         'bt_volumeto': 'float', 'bt_target': 'float'})  
  
target = df_train_matrix['bt_target']  
# mixed data (market and tweets)  
data = df_train_matrix[['neu_mean', 'neg_mean', 'pos_mean', 'comp_mean', 'pol_mean', 'bt_close', 'bt_open', 'bt_high', 'bt_low', 'bt_volumeto']]  
# market data  
data = df_train_matrix[['bt_close', 'bt_open', 'bt_high', 'bt_low', 'bt_volumeto']]  
# tweets data  
data = df_train_matrix[['neu_mean', 'neg_mean', 'pos_mean', 'comp_mean', 'pol_mean']]
```

Separação dados treinamento e testes

```
X, V_test, Z, Z_test = train_test_split(data, target, test_size=0.3, random_state = 1)
```

Criação do modelo de Machine Learning

Treinamento da MLP

```
def neural_train(data, target):
    V, V_test, Z, Z_test = train_test_split(data, target, test_size=0.3, random_state = 1)

    scaler = preprocessing.StandardScaler()
    #Fit only over test data
    scaler.fit(V)
    V = scaler.transform(V)
    V_test = scaler.transform(V_test)

    random.seed = time.time()
    max_acc = -1
    best_mlp = None
    max_pre = -1

    for i in range(50):
        # mlp = MLPClassifier(solver='adam', alpha=0.0002, learning_rate_init=0.00001, max_iter=500, early_stopping=True, hidden_layer_sizes=(55, ), activation='tanh', random_state=1)
        mlp = MLPClassifier(solver='lbfgs', alpha=0.0002, learning_rate_init=0.00001, max_iter=500, early_stopping=True, hidden_layer_sizes=(30, ), activation='tanh', random_state=1)
        # mlp = MLPClassifier(solver='sgd', alpha=0.0002, learning_rate_init=0.00001, max_iter=500, early_stopping=True, hidden_layer_sizes=(55, ), activation='tanh', random_state=1)

        mlp.fit(V, Z)

        Z_predict = mlp.predict(V_test)

        acc = accuracy_score(Z_test, Z_predict)

        pre = precision_score(Z_test, Z_predict)

        if acc > max_acc:
            max_acc = acc
            best_mlp = mlp

        if pre > max_pre:
            max_pre = pre
```

Criação do modelo de Machine Learning

Treinamento da MLP, Random Forest e SVM

```
def neural_train(data, target):
    V, V_test, Z, Z_test = train_test_split(data, target, test_size=0.3, random_state = 1)

    scaler = preprocessing.StandardScaler()
    #Fit only over test data
    scaler.fit(V)
    V = scaler.transform(V)
    V_test = scaler.transform(V_test)

    random.seed = time.time()
    max_acc = -1
    best_mlp = None
    max_pre = -1

    for i in range(50):
        # mlp = MLPClassifier(solver='adam', alpha=0.0002, learning_rate_init=0.00001, max_iter=500, early_stopping=True, hidden_layer_sizes=(55, ), activation='tanh', random_state=1)
        mlp = MLPClassifier(solver='lbfgs', alpha=0.0002, learning_rate_init=0.00001, max_iter=500, early_stopping=True, hidden_layer_sizes=(30, ), activation='tanh', random_state=1)
        # mlp = MLPClassifier(solver='sgd', alpha=0.0002, learning_rate_init=0.00001, max_iter=500, early_stopping=True, hidden_layer_sizes=(55, ), activation='tanh', random_state=1)

        mlp.fit(V, Z)

        Z_predict = mlp.predict(V_test)

        acc = accuracy_score(Z_test, Z_predict)

        pre = precision_score(Z_test, Z_predict)

        if acc > max_acc:
            max_acc = acc
            best_mlp = mlp

        if pre > max_pre:
            max_pre = pre
```

```
# Random Forest
rf = RandomForestClassifier(n_estimators=1000)
rf.fit(V,Z)

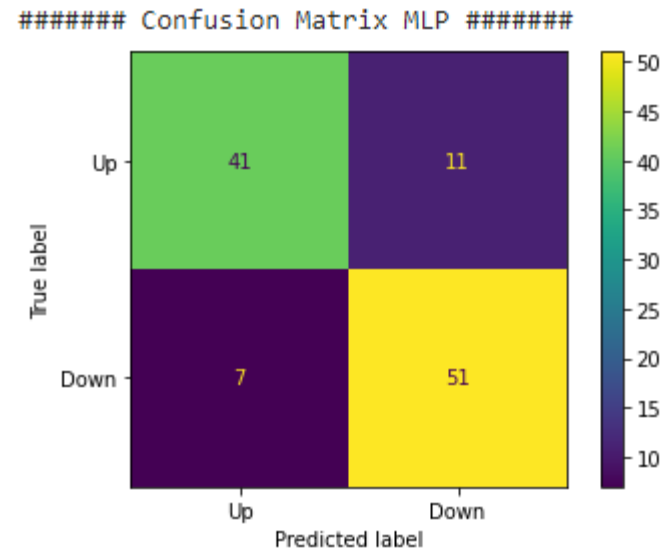
Z_predict = rf.predict(V_test)
```

```
# SVM
svm = SVC(kernel='rbf')
svm.fit(V,Z)

Z_predict = svm.predict(V_test)
```

Interpretação de Resultados

Matriz de confusão da melhor MLP e validação cruzada



Interpretação de Resultados

- Resultado com dados Bitcoin

	Precision	Accuracy	Recall	F1
PMC	0,81	0,77	0,77	0,78
Random Forest	0,72	0,68	0,68	0,7
SVM	0,72	0,63	0,55	0,62

- Resultado com dados Twitter

	Precision	Accuracy	Recall	F1
PMC	0,52	0,5	0,51	0,52
Random Forest	0,53	0,51	0,65	0,58
SVM	0,57	0,56	0,68	0,62

- Resultado com dados Twitter e Bitcoin











	Precision	Accuracy	Recall	F1
PMC	0,82	0,83	0,87	0,85
Random Forest	0,6	0,58	0,6	0,6
SVM	0,58	0,58	0,72	0,64

A rede neural PMC se mostrou o melhor algoritmo para realizar previsões com estes dados.

Mas a validação cruzada da PMC...

```
##### Cross Validation MLP #####  
Cross Validation Precision: 0.5123123123123123  
Cross Validation Accuracy: 0.5123123123123123  
Cross Validation Recall: 1.0  
Cross Validation F1: 0.6774675324675326
```

Apresentação de Resultados

PREDICTION TASK  <p>A tarefa de <u>machine learning</u> é o de tentar prever o que vai acontecer com o mercado do Bitcoin no dia seguinte. Será que o mercado vai subir ou vai cair.</p> <p>Dos algoritmos testados o que obtivemos melhor resultado foi o <u>Perceptron</u> com Múltiplas Camadas, chegamos a uma precisão de 82%.</p>	DECISIONS  <p>O resultado do modelo é predição da variação diárias do mercado de Bitcoin para permitir ao <u>trader</u> ter mais confiança nas operações de compra ou venda da moeda digital.</p>	VALUE PROPOSITION  <p>A intenção é de fornecer ao <u>trader</u> uma forma mais confiável para que ele possa realizar as transações com o Bitcoin de forma mais assertiva, ajudando a ter mais segurança nas suas transações.</p>	DATA COLLECTION  <p>Novos dados podem ser obtidos via API do Twitter via contrato com esta empresa e para os Bitcoins pode ser obtido de forma gratuita via API <u>Rest</u> da Cripto Compare.</p>	DATA SOURCES  <p>As fontes de dados para realização deste trabalho foram <u>dados</u> do Twitter e do mercado de Bitcoin do ano de 2021.</p> <p>Os dados do <u>twitter</u> foram obtidos a partir de web <u>scrapping</u> e os do Bitcoin via API <u>Rest</u> da Cripto Compare.</p>
IMPACT SIMULATION  <p>Ao verificar o modelo usando a validação cruzada verificamos que o modelo apresentou muito baixa precisão, em torno de 51%, com uma acurácia de igual valor, um recall de 100% e f1 de 67%.</p>	MAKING PREDICTIONS  <p>As predições são realizadas logo após a obtenção e os tratamentos nos dados e pode ser feita de forma bem rápida.</p>	BUILDING MODELS  <p>Foram testados o <u>Perceptron</u> com Múltiplas Camadas, e os algoritmos <u>Random Forest</u> e <u>Support Vector Machines</u>. O melhor modelo foi o PMC que foi utilizado nas predições.</p>		FEATURES  <p>As variáveis preditoras X são: <u>'neu_mean'</u>, <u>'neg_mean'</u>, <u>'pos_mean'</u>, <u>'comp_mean'</u>, <u>'pol_mean'</u>, <u>'bt_close'</u>, <u>'bt_open'</u>, <u>'bt_high'</u>, <u>'bt_low'</u> e <u>'bt_volumeto'</u>.</p> <p>E a variável alvo Y é: <u>'bt_target'</u></p>
MONITORING  <p>O monitoramento do modelo foi realizado por meio da verificação da medida de precisão dos modelos testados.</p>				

Modelo Canvas proposto por Dourad.

Pós-graduação Lato Sensu em Ciência de Dados e Big Data
Trabalho de Conclusão de Curso

MODELO PREDITIVO DE VARIAÇÃO DA
CRIPTOMOEDA BITCOIN

Aluno: **Jean Carlos Sousa Silva**

Belo Horizonte
2022